Secure and QoS-Managed Information Exchange Between Enterprise and Constrained Environments

Partha Pal, Michael Atighetchi, Nathaniel Soule, Vatche Ishakian and Joseph Loyall
Raytheon BBN Technologies
Cambridge, MA, USA
{ppal, matighet, nsoule, vishakia, jloyall}@bbn.com

Robert Grant and Asher Sinclair
US Air Force Research Laboratory
Rome, NY, USA
{Robert.Grant.11, Asher.Sinclair.1}@us.af.mil

***Abstract*— Mobile devices performing mission-critical functions at the *tactical edge*, such as those employed by first responders, military personnel, and law enforcement, operate in environments that are vastly different from *enterprise* computing environments. In spite of the differences in resource availability, threat models, vulnerabilities, information formats, and communication protocols, there is a great advantage to (and great demand for) enabling information exchange between the tactical edge and enterprise environments. Creating a specialized mobile version of each desired service that incorporates an appropriate level of security protection and quality of service (QoS) for the tactical users is one possibility. Such an approach is not cost effective, however, as the market for a given tactical application is small compared to the commercial user base for mobile applications and services. Furthermore, the need for information or services from the enterprise by tactical users can be too ad hoc and time critical, e.g., during disaster response, to support developing a specialized version. Finally, service specialization for mobile web access covers only one of multiple information dissemination and access patterns that arise in tactical operations. This paper presents the design and a prototype implementation of a gateway solution that provides secure tactical-enterprise information exchange and handles the differences in resource availability, QoS requirements, communication formats, and protocols.**

Keywords—gateway; middleware service; tactical and enterprise environments; quality of service; security; information management

I. Introduction

Operating conditions, available resources, and potential threats all differ significantly between mobile and enterprise computing environments. Enterprise systems enjoy high bandwidth, redundant connections, robust power supplies, virtually unlimited storage, fast processors, and a level of physical security in their surroundings. In contrast, mobile computing is often limited to low bandwidth radio communications, short-lived batteries, small on-device storage, less powerful processors, and smaller displays. As a result, mobile devices often cannot run intrusion detection systems or virus scanners, receive and store large files, display high resolution video and images or run strong encryption to the same extent as their enterprise counterparts. These differences are even more pronounced in the context of tactical missions such as disaster response, humanitarian support in conflict zones, law enforcement, and counter terrorism operations where handheld and mobile devices communicate with remote services over tactical radios in place of, or in addition to, the commercial cellular and Wi-Fi networks. Users in tactical environments also face a high level of dynamism, physical threats, and stress beyond the casual mobile computer user.

Information exchange between tactical systems and enterprise services during mission operation can enable greater situational awareness and empowerment for the tactical user. For example, first responders at a disaster scene could access diagnostic software or medical records maintained at enterprise healthcare information systems or border protection personnel could use analysis services hosted in the enterprise. Common approaches to enabling information exchange between commercial mobile devices and enterprise environments include mobile versions of web portals, device-based access permissions, and out-of-band wire-line synchronization. These are insufficient for tactical to enterprise interactions for several reasons. First, these concentrate on the tactical device to enterprise path, whereas many situations involve two-way interactions where the tactical devices need to host services for enterprise clients in addition to the usual web browsing and content downloading interactions. Second, solutions that involve staging or offloading assets into mobile devices are risky in tactical environments, since the devices can be lost, stolen, or otherwise fall into the hands of an untrusted party. Third, tactical-enterprise interactions must not incur unpredictable delays during time-critical operations (such as law enforcement or emergency response). Finally, creating specialized versions of enterprise services for tactical access, like is commonly done for commercial mobile applications, is challenging economically (the user and developer base is too small to sustain a viable marketplace) as well as technically (dynamically arising needs for tactical access leave little time for new development).

In this paper, we present an alternate approach, a gateway-based solution that enables safe and efficient two-way information exchange between tactical and enterprise environments, with little or no change to the applications or servers in those environments. The design and prototype im-

This research is funded by the United States Air Force Research Laboratory under Contract No. FA8750-12-C-0071. Approved for public release. Case No. 88-ABW-2013-2466.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **2014**	2. REPORT TYPE	3. DATES COVERED **00-00-2014 to 00-00-2014**

4. TITLE AND SUBTITLE **Secure and QoS-Managed Information Exchange Between Enterprise and Constrained Environments**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Raytheon BBN Technologies,10 Moulton Street,Cambridge,MA,02138**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
17th IEEE Symposium on Object/Component/Service-oriented Real-time Distributed Computing, Reno, NV, June 10-12 2014.

14. ABSTRACT
Mobile devices performing mission-critical func-tions at the tactical edge, such as those employed by first re-sponders, military personnel, and law enforcement, operate in environments that are vastly different from enterprise computing environments. In spite of the differences in resource availability, threat models, vulnerabilities, information formats, and commu-nication protocols, there is a great advantage to (and great de-mand for) enabling information exchange between the tactical edge and enterprise environments. Creating a specialized mobile version of each desired service that incorporates an appropriate level of security protection and quality of service (QoS) for the tactical users is one possibility. Such an approach is not cost ef-fective, however, as the market for a given tactical application is small compared to the commercial user base for mobile applica-tions and services. Furthermore, the need for information or services from the enterprise by tactical users can be too ad hoc and time critical, e.g., during disaster response, to support devel-oping a specialized version. Finally, service specialization for mobile web access covers only one of multiple information dis-semination and access patterns that arise in tactical operations. This paper presents the design and a prototype implementation of a gateway solution that provides secure tactical-enterprise information exchange and handles the differences in resource availability, QoS requirements, communication formats, and protocols.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**9**	

plementation of the Secure Tactical-Enterprise Gateway (STEG) address the challenges described above and offer a blueprint for unified management of the security and QoS asymmetry between tactical and enterprise environments at an acceptable overhead. STEG achieves this goal by combining traditional security techniques like authentication and authorization with the innovative use of *crumple zones* [1] and *protocol transformation*, complemented by multi-level prioritized queuing at the egress for QoS management. The gateway approach ensures that a unified, managed, and auditable set of checks, optimizations, and transformations are employed for all tactical to enterprise and enterprise to tactical communication.

STEG is designed as a dual-homed physical or virtual appliance with one network interface connected to the tactical environment and the other to the enterprise. As a result of the gateway approach, new systems and devices can be added without modification and lightweight mobile devices can take advantage of the heavier weight processing capabilities, more robust logging, auditing, aggregated threat detection, and remediation managed at the gateway, making the STEG solution portable, and quick and easy to deploy. STEG's design is customizable to accommodate various security and QoS policies as needed for different situations and deployments. New tactical or enterprise message formats, cross-environment authorization policies, security inspections, and QoS policies can be easily accommodated by adding pluggable components and editing XML-based policies.

The remainder of this paper is organized as follows. Section 2 describes a use case highlighting the features and benefits of STEG. Section 3 provides an overview of the STEG solution, describing the architecture and design choices, supported capabilities, and implementation details. Section 4 describes evaluation results, followed by related work in Section 5. Section 6 concludes with a summary and discussion of future work.

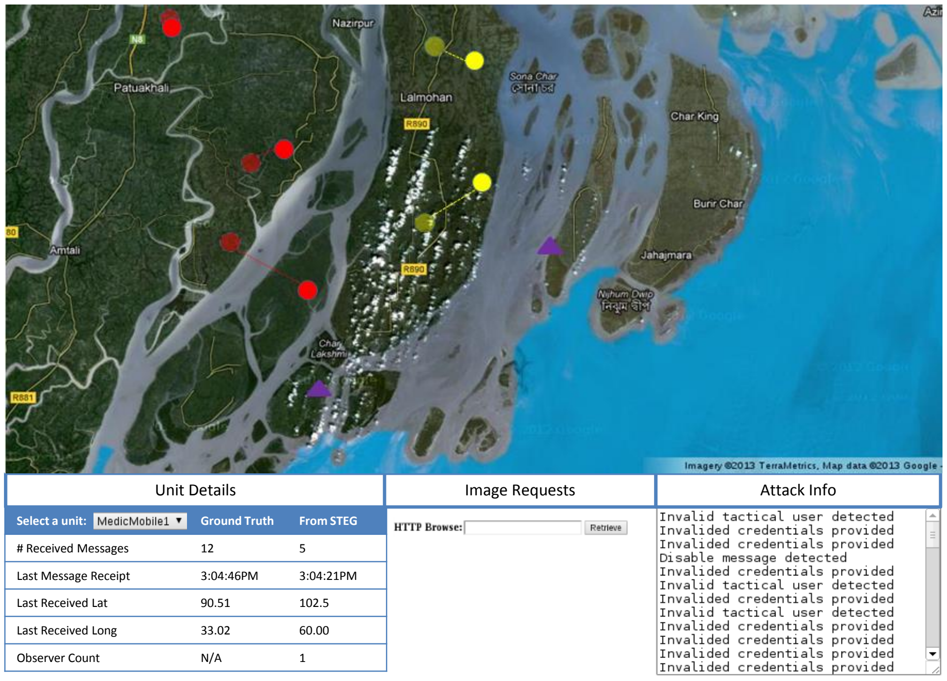
II. Prototypical Use Case

Consider a situation where international agencies and non-governmental organizations (NGOs) are collaborating with a host government in a disaster area which is also home to an armed domestic insurgency. The NGOs have pre-established bases in the region that are now threatened by a cyclone. In addition to relief and medical services, aid workers and displaced people need protection from the insurgents. Mobile field units from the host country and international volunteers use radios that are short range and use different standards. A command center is established at the enterprise network of the host government for coordination. Satellite imagery of the flooded area and insurgent movement are made available to the command center by international collaboration. Both field units and the enterprise side services need a way to effectively and efficiently publish and consume critical information.

The tactical radio links have less bandwidth compared to links on the enterprise side, and cannot receive the large high resolution images obtained through the network of international entities. Some of these images may be of higher priority than others. STEG's QoS processing addresses this issue by prioritizing the data and shaping it to fit the resources of the radio links. For example, STEG will reduce the size of the original image before forwarding it to the tactical client, who can request the original full size version if needed.

Cyber-attacks by the insurgents (and adversaries that frequently target NGOs to gain entry to other systems [6]) may attempt to disrupt the aid operation and exfiltrate sensitive information. STEG's security-focused processing mitigates these risks. For example, STEG will block unauthenticated clients and unauthorized data flows, detect and remove viruses, and absorb and automatically recover from the effects of unknown attacks while shielding the downstream clients.

This use case serves as the backdrop for STEG testing, evaluation, and demonstration. A snapshot of this mission's execution is shown in STEG's demonstration viewer in Fig. 1. Movements of aid agencies (yellow dots) and insurgents (red dots), and static NGO quarters (purple triangle) are displayed on a map. A panel at the bottom shows event statistics and security-related information. The actual location of tracked entities as well as their location as received by data consumers (who can receive the information only through STEG) are displayed, and the difference between these values represented visually (e.g., update lag and distance between the nodes at a given time). For comparison, what a legitimate consumer would experience without STEG can also be shown in this



Select a unit: MedicMobile1	Ground Truth	From STEG
# Received Messages	12	5
Last Message Receipt	3:04:46PM	3:04:21PM
Last Received Lat	90.51	102.5
Last Received Long	33.02	60.00
Observer Count	N/A	1

```
Invalid tactical user detected
Invalided credentials provided
Invalided credentials provided
Disable message detected
Invalided credentials provided
Invalid tactical user detected
Invalided credentials provided
Invalid tactical user detected
Invalided credentials provided
Invalided credentials provided
Invalided credentials provided
Invalided credentials provided
```

Fig. 1. STEG Operational View

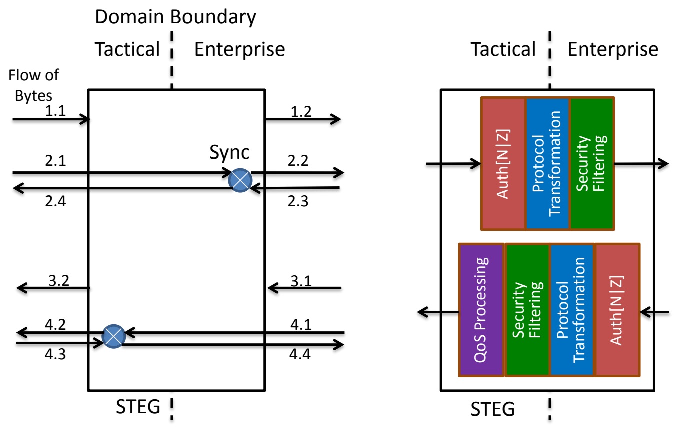
viewer, showing the data loss or delay due to unmanaged tactical network congestion. In the event of a cyber-attack, security events observed are presented in the bottom panel. Successful absorption of attack effects and subsequent recovery of information flow is shown as occasional differences between the (pre-STEG) ground-truth and (post-STEG) received-truth positions.

III. The STEG Solution

A. *Concept of Operation and Architecture Overview*

STEG operates by taking in messages on one interface, performing security-focused and QoS-focused processing, and sending the messages out another interface. This basic interaction path through STEG is enabled by appropriate ingress mechanisms that listen for new connections and egress mechanisms that disseminate information in a format and manner expected by the receiving side (which in many cases is not the same as the input format/protocol). Many of the desired information flows to be exchanged between tactical and enterprise environments are asynchronous and involve one-way messages, e.g., publish-subscribe (see interactions 1.1-1.2 and 3.1-3.2 in Fig. 2a). Other desired cross-environment interactions are synchronous, i.e., a request from one side to the other blocks and needs to be synchronized with the response(s) going in the other direction (see interactions 2.1-2.4 and 4.1-4.4 in Fig. 2a), requiring additional synchronization within STEG (represented by the crossed oval in Fig. 2a). The STEG prototype can handle interactions from both these categories, specifically supporting publish, subscribe, disseminate (typical information management operations), and request/reply (such as HTTP GET/POST requests).

Fig. 2b shows the organization of STEG's basic functional processing blocks in both directions. The authentication (AuthN) and authorization (AuthZ) block authenticates incoming messages and checks whether information contained in the messages are authorized to be released to the other side. The protocol transformation block transforms incoming messages into an internal format partly to disrupt protocol-specific attacks and partly to make it easier to map to a destination format (which happens at the QoS processing block). The security filtering block inspects the submitted content, validates the



a) One-way flows and RPCs both ways

b) Main Functionality

Fig. 2. STEG Concept of Operation

input, and ensures continuity and integrity of legitimate information flows, while preventing unauthorized and malicious ones. The QoS processing block prioritizes, shapes, and filters messages to accommodate the resource restrictions and format/protocol requirements of the receiving side. QoS processing is performed just before messages egress out of STEG to have the best impact on delivery. Authentication and authorization are performed close to the ingress point so that illegitimate flows are cut as early as possible, thus conserving resources. Protocol transformation to the internal format (XML data over HTTP) is performed as soon as messages are authenticated and authorized, and the same format is used for all intra-STEG communication until the messages are transformed again to the destination format at the egress point. Unlike authentication and protocol transformation functionalities, which are essential for information exchange in both directions, QoS control functionality is more important for tactical egress because network and CPU resources are usually less plentiful in the tactical side than in the enterprise. Being a gateway, STEG cannot exert any control on the sending applications beyond the refusal of connections and the congestion control or throttling provided by the underlying wire or radio protocol, but it can control what it injects into a resource constrained network, at what rate, and in what order.

Although Fig. 2b shows the enterprise to tactical (and the reverse) interaction as a single pipeline of functional blocks, STEG is organized as multiple pipes of components and services dedicated to the different types of directional flows for isolation, redundancy, and load balancing. Even though inter-component interactions within STEG happen over HTTP, the flow- and direction-specific isolation provides containment, i.e., the ill effects of a compromised STEG component are limited to the disruption of the affected pipeline, while other components and flows continue unabated. Key STEG components also run under process protection domains enforced by customized SELinux [18] and JVM security policies [8]. Finally, frontline STEG components that are exposed to incoming traffic before security-processing, and hence are more vulnerable to corruption and failures, are monitored by *watchdogs*, which detect corruption and failures and restart the components when needed.

As shown in Fig. 3, STEG consists of the following three architectural subsystems that together enable the functionality stated above:

- *Façade Subsystem*: contains connectors that accept connections and data from one side (i.e., enterprise or tactical) that is destined for the other. The connectors conduct authentication, authorization, and protocol and format transformation.
- *Crumple Zone Subsystem*: run security checks, inspection, emulation, and other techniques to absorb and block malicious traffic from reaching the target environment.
- *Service Subsystem*: responsible for making STEG appear as a transparent gateway to either side, and include proxies for one-way (e.g., publish-subscribe) and two-way (e.g., browse) interactions and a QoS-enabled dissemination (QED) service [11]. The QED service delivers data to intended recipients, but utilizes priority queuing, data

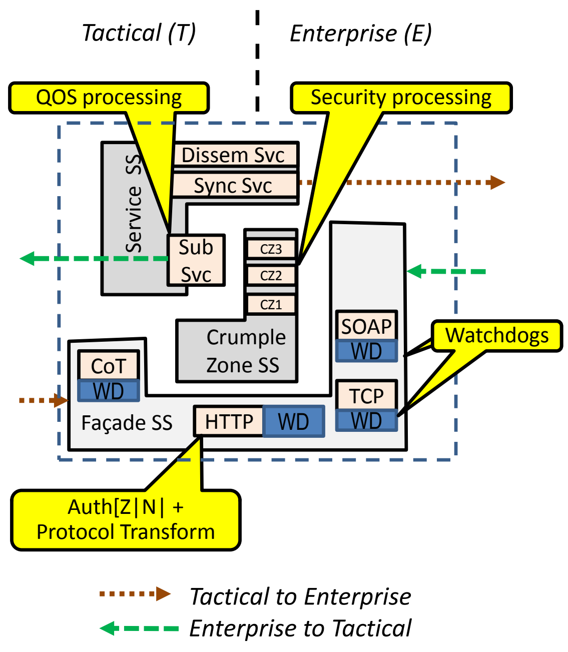


Fig. 3. STEG Architecture

filtering, and format and protocol transformations to make the dissemination suitable for the target environment.

STEG uses the Logback distributed logging framework to collect streams of log events from various STEG components reporting key lifecycle and operational events. These events can be exported outside of the STEG appliance for further processing by auditing and metrics processing frameworks such as Metrinome [2] and for extracting and visualizing health and operational metrics. All unnecessary OS services and features are removed or disabled in the STEG appliance. Computation in STEG is run in well-defined *network* and *process protection domains.* For the network protection domain, STEG uses separate network interfaces for the tactical and enterprise sides, and employs separately configured firewalls. STEG can optionally be configured with Single Packet Authorization [16] where STEG's network ports are not accessible without appropriate cryptographic credentials. Key STEG processes such as the connectors run under SELinux policies that control the OS resources that they can access. Finally, Java components are subjected to JVM security policies.

B. The Façade Subsystem

The STEG façade is the home of the connectors, their watchdogs, and the authentication and authorization services.

Connectors. The connectors are responsible for listening to incoming connection requests and performing initial protocol transformation that maps incoming messages into a STEG internal messaging format (XML over HTTP). Connectors also invoke the authentication and authorization services.

The STEG façade presents a variety of connectors that accept different types of service requests over a diverse set of protocols. The reason for this diversity is that protocols used on the tactical side are often different from what is used in the enterprise side. For example, SOAP over HTTP is commonly used in the enterprise side whereas tactical systems use protocols like *Cursor on Target (CoT)* [7][17]. In order to enable tactical clients to access enterprise services (and similarly, enterprise clients to access tactical services) without changing the clients or services, connectors accept the service requests as they are and map them to requests on the desired resource or service at the other side, internally managing the transformation of requests and responses from one protocol format to another.

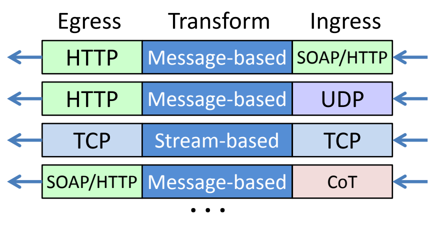


Fig. 4. Façade Connectors

As shown in Fig. 4, each connector consists of an ingress building block that is exposed to external clients, a protocol format transformation process, and an egress building block that is exposed to downstream internal STEG components. We have developed a library of ingress and egress blocks handling various protocols (including HTTP, SOAP over HTTP, UDP, TCP, and CoT). The connector design is flexible to make it easy to construct new types of connectors by combining ingress and egress blocks as long as they are both stream-based or both message-based (see Fig. 4). This composable connector structure not only enables serving different types of connection requests, but also facilitates protocol transformation. Since STEG internal components interact with each other using HTTP, the most common type of transformation in the façade is from other protocols to HTTP. Unless the intended target of the request also uses HTTP, STEG must perform another transformation when it hands the request off to the target network. This transformation is handled by the STEG service that is responsible for the handoff, such as the QED service. Protocol transformation also serves a security purpose; i.e., transformations may disrupt specific attacks that attempt to exploit protocol-specific vulnerabilities by using specially crafted messages.

Watchdogs. Connectors are directly exposed to incoming messages from outside and have a high risk of getting compromised. Therefore connectors are monitored by watchdogs (WD in Fig. 3), specialized components than can kill and restart suspected connectors that exhibit aberrant behavior. In its simplest form, a watchdog detects process termination and performs a restart. The STEG watchdogs go beyond simple crash failure detection in several ways. First, STEG continually monitors thread-level CPU usage to protect against malicious code that executes in the processes and threads that handle input data. STEG maintains a map of <thread ID, source IP address> for those threads handling incoming connections. If the watchdog detects excessive CPU usage for a given process, it replaces the process with a fresh copy and identifies the thread and IP address associated with the abnormal processing. The watchdog can also modify firewall rules to automatically block the offending IP address. Second, monitored processes can be configured to send periodic heartbeats to their corresponding watchdog. A continued lack of receipt of these messages indicates that the process is unable to operate normally. The watchdog then replaces the suspected process with a fresh copy. Third, all STEG processes produce log messages that watchdogs can monitor for observable anomalous behavior. When a watchdog is configured, it is given a regular expression that will match log messages produced by its monitored process under normal operation. Since low or idle loads could cause lack of log messages, watchdogs periodically trigger processing (and log message generation) of the monitored process through its service interface. Lack of expected log messages is interpreted to mean either the process is not performing its expected tasks or not doing so in a timely manner.

Authentication Service. STEG validates the identity of processes that request or provide services and that produce or receive information through STEG. STEG's authentication service supports various modes of authentication, from explicitly identified trusted devices (e.g., granting access to trusted IP addresses) to using X.509 certificates available via Transport Layer Security (TLS) [5]. The STEG prototype employs a TLS 1.2 cipher-suite and configuration (i.e., AES-256-CBC with JSSE Strict Mode settings and 2048 bit keys) that are not vulnerable to recently discovered attacks [3].

Authenticating across the tactical-enterprise boundary is challenging because these environments differ in the way they manage their resources and identities. For example, one enterprise server might use username and passwords while another might use smartcards, whereas tactical applications may rely on pre-loaded keys. There is no notion of a universal identity that is valid in all environments. To address this challenge, STEG maps the identity of an authenticated ingress request into another identity (given to STEG) that is valid on the target side before the request egresses the gateway. Fig. 5 shows the STEG authentication and authorization architecture. STEG provides a library for defining identity mapping rules (*Identity Map Policy*) in XML. The *IdMapper* replaces the original credential by the credential of the mapped identity in each request that egresses STEG. Mappings can be 1-to-1 or many-to-1.

Authorization Service. STEG uses Attribute Based Access Control (ABAC) [10] to determine whether to process authenticated requests further or to reject them. STEG's authorization service consults a built-in *Attribute Authority (AA)* to look up the attributes of an authenticated identity and

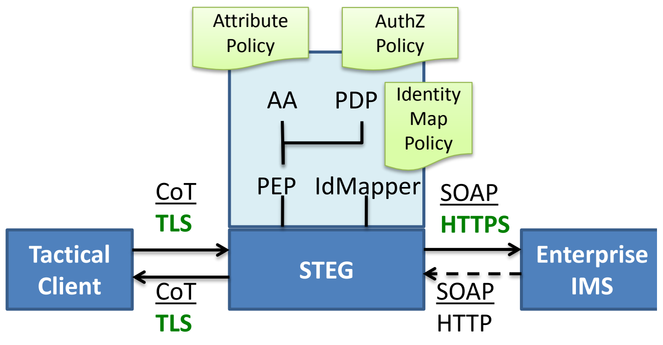


Fig. 5. Authentication and Authorization architecture of STEG

check them against the *Attribute Policy*. Once attributes are determined, the *Policy Decision Point (PDP)* evaluates the request against access control rules, defined in a separate *Authorization (AuthZ) Policy*. The *Policy Enforcement Point (PEP)* denies requests unless explicitly allowed by the PDP.

C. *The Crumple Zone Subsystem*

Most automobiles today are equipped with a *crumple zone* – an area of the vehicle that absorbs the impact resulting from collisions and shields the human passengers. In order to be an effective gateway that absorbs the shock of attacks from one side on the other, all interactions going through STEG are subjected to security-focused processing that mimics the physical concept of a *crumple zone* [1]. The STEG crumple zones subsystem is essentially a collection of individual crumple zones, one for each connector, handling the interactions accepted by that connector. Within a crumple zone, all incoming data is split, with one copy escrowed in a buffer while the other is subjected to security-focused inspection, as shown in Fig. 6. Escrowed data is released only if all security checks are satisfied.

A configurable collection of mechanisms participate in the security-focused inspection. Where possible, these mechanisms operate in parallel. Some of the mechanisms operate in a streaming mode, signaling the escrow to release chunks of data incrementally. Others need to read in the whole message before processing, sending a release signal when the entire message has been read and processed. The escrow releases a chunk of escrowed data only if all inspection mechanisms deem that the chunk is safe for release. In our prototype implementation, security focused processing of the data passing through STEG covers the following: 1) *HTTP request format verification* inspects an HTTP message to ensure that it meets the specification; 2) *Virus scanning* runs the data flowing through a virus scanning tool (e.g., ClamAV [4]); 3) *Size limiting* enforces a (configurable) size threshold, above which warnings are issued or messages are blocked; 4) *Rate limiting* enforces a (configurable) rate threshold, if a connection sends data faster than the threshold, messages are dropped; and 5) *Canary execution* partially executes requests by deserializing Java objects to protect against the execution of arbitrary code that is possible during deserialization of cleverly crafted serialized objects. The canary mechanism absorbs such attacks because the attack affects the canary, not the target system.

The inspection mechanisms are able to detect the presence of malicious content directly (i.e., by scanning the bytes) or

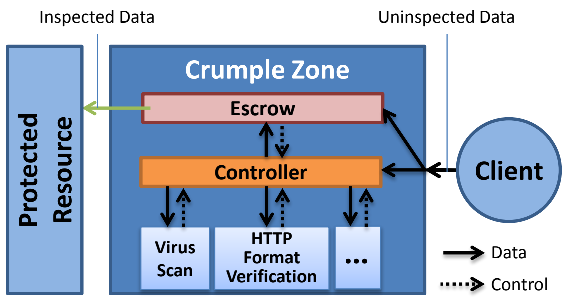


Fig. 6. Crumple Zone Architecture

indirectly (i.e., by triggering unexpected behavior as in the Canary Execution). The components performing security inspection are isolated from the rest of the system and are monitored by watchdogs as described earlier. A compromise in such a component is an indication that a potentially dangerous flow was prevented from flowing downstream, including to the intended target.

D. The Service Subsystem

To make STEG a transparent gateway, STEG includes services that handle the common operations of expected clients such as publish, subscribe, disseminate and browse requests without breaking (1) the synchronous and asynchronous nature of the interaction, (2) the QoS profile, or (3) the expected protocol format of the destination environment. The *Sync* service handles synchronization of browse requests and responses. The *QED* service utilizes priority queuing, data filtering, and format and protocol transformations to make the dissemination suitable for the target environment. The *Subscription* service is responsible for mapping subscription requests from the source environment to the destination environment. Note that a separate *Publication* service is not needed because for tactical to enterprise publication, the destination format and the STEG internal (HTTP) format are the same, and for enterprise to tactical publication, the payload can be large with respect to the resource constrained tactical network, and hence the publication messages are routed through the QED service for QoS management.

The QED Service. The QED service is responsible for delivering information to the intended recipient. If a tactical client's request matches an information object published on the enterprise side, the object may be larger than what the tactical network can handle. The publication service, trying to deliver content received from the enterprise side to a tactical-side service may suffer the same issue. Similarly, the Sync component may obtain a large amount of data in its interaction with an enterprise web server that is destined for a tactical browser client. The QED service delivers information by considering the type of data, the perceived load on the destination, and policy prescribed via priority, deadline, and criticality.

Internally, the QED service operates by manipulating the various queues it maintains for handling egress traffic based on applicable QoS policy. QED supports several queue manipulation techniques including *priority filtering*, *data shaping*, *deadline enforcement* and *replacement*. For *priority filtering*, the QED service maintains a set of priority queues and delivers higher priority content in preference to lower priority content, based on a configurable strict or weighted fair strategy. *Data shaping* involves transforming the data to better fit the available bandwidth or the display characteristics of the receiving device. The shaping primitives currently supported include image resizing, compression, eXtensible Stylesheet Language Transforms (XSLT) on XML data, and payload filtering (when metadata alone can meet client needs).

Mission-critical information may become stale after a certain period of time, and in some missions not receiving the data may be preferable to receiving stale data. STEG's QoS policy can indicate message types that have deadlines and *deadline enforcement* within QED will drop stale messages of those types. Certain classes of information have semantics such that the receipt of a later message invalidates the need for earlier messages of the same type. For example, in a stream of periodic messages indicating the current weather conditions, one is likely to only be concerned with the most recent data. The *replacement policy* supported by the QED service provides a way to annotate messages that are *replaceable*, and drops all but the latest if multiple replaceable messages of the same type are backed up in the dissemination queue.

In radio-based tactical environments, devices may become unreachable for a while before rejoining the network. Terrain, weather, distance, and attacks can all lead to temporary disconnects from the larger network. STEG handles intermittency by intelligently connecting lower-level network failure handling with the management of dissemination queues at the QED service. Specifically, the QED service includes pluggable transmission failure handlers with various flavors of exponential back-off and retry within the applicable deadline, priority, or replacement policies.

The Sync Service. STEG needs to make synchronous interactions such as web browsing look like two-way communications to the requester at the STEG ingress and the destination service at the STEG egress. In both cases, the connection needs to be held open until the response becomes available. At the STEG ingress, the connector services synchronous service requests, but the STEG connector is a long way from the STEG egress and is ill-suited to handle the synchronization with the destination service because of the escrowing and security-focused inspection that needs to happen on both the request and the response. The Sync service, situated near the STEG egress, provides the synchronous interaction with the end destination in the egress side.

The Subscription Service. The Subscription Service enables information requests (i.e., subscriptions) from clients in one domain to information management services (IMS) that manage and serve subscriptions in another. The Subscription service is designed as a composition of two plugin modules, one each for the type of systems/clients served in the two environments bridged by the gateway in order to handle any asymmetries between the ways subscription requests are handled in the tactical and enterprise domains.

IV. Evaluation

The evaluation of STEG reported here focused on (1) testing whether the initial (unoptimized) STEG prototype can provide continuous cross-environment interactions under attack and (2) understanding any incurred performance overhead. QED capabilities have been evaluated extensively before in [13].

A. Experimental Setup

Fig. 7 shows the experimental configuration. STEG runs on the middle Virtual Machine (VM) configured with one virtual CPU and 2GB of RAM, connected to virtualized (emulated) tactical and enterprise networks on each side. An enterprise IMS runs on a VM on the enterprise side. Producer and consumer applications on both the enterprise and tactical sides interact with this IMS. The STEG VM was based on VirtualBox 4.1.18, running Java 7 on Fedora 14 as the guest OS. On

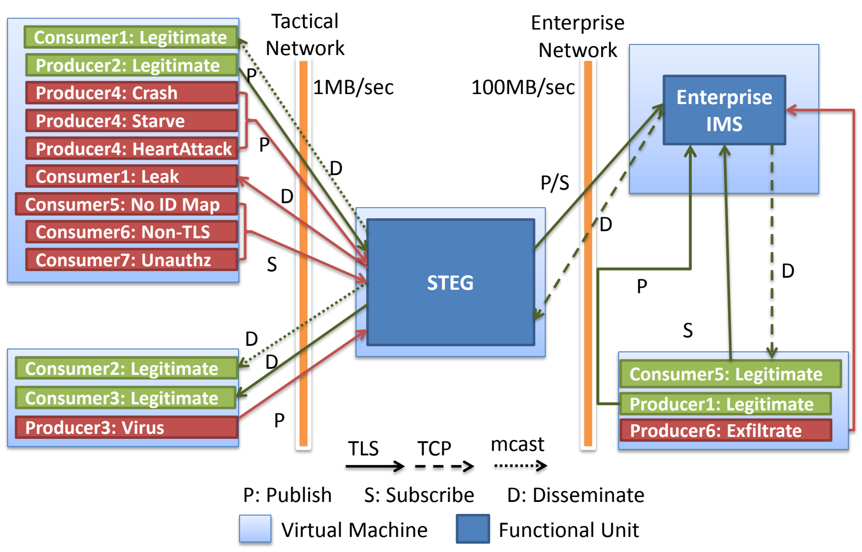


Fig. 7. System under test for experimental evaluation of STEG

the enterprise-to-tactical dissemination path, tactical consumers (Consumer 1, 2 and 3) seeking information managed by the enterprise IMS subscribe with STEG to receive matching Information Objects (IOs) via unicast or multicast. An enterprise producer (Producer 1) publishes IOs to the IMS, which evaluates the IOs against registered subscriptions. IOs that match subscriptions registered by tactical users are sent to STEG, which disseminates the IOs to the tactical consumers. On the tactical-to-enterprise dissemination path, an enterprise consumer (Consumer 4) receives IOs published by tactical Producer 2. STEG enables the publication to the enterprise IMS by a tactical producer, the enterprise consumer simply subscribes with the (local) enterprise IMS. In addition to normal producers and consumers, the experiments have malicious applications (red rectangles) on both sides that launch attacks across STEG. Bandwidth to STEG from the tactical and enterprise sides are set at 1 Mbps and 100 Mbps, respectively, to emulate realistic networking conditions.

B. *Functioning through Attacks*

In this test, malicious producers and consumers (shown in red in Fig. 7) continuously injected attack behaviors and effects in parallel with benign requests from legitimate producers and consumers. The malicious interactions included sending messages that crash or stall connectors, cause connectors to hog the STEG CPU and starve other STEG processes (from Producer 4); sending unauthorized IOs from enterprise to tactical as subscriptions, and connecting to a TLS port with a regular TCP connection (from Producer 6); connecting with an identity from the tactical side that does not map to any defined enterprise identity (from Consumer 5); subscribing to unauthorized topics (Consumer 7); and propagating a virus (from Producer 3). In a set of experimental runs each lasting approximately 30 minutes, STEG continued to provide unimpeded service to legitimate clients, despite the continuous and sustained parallel attacks. Due to the fact that legitimate and attack interactions were sent by asynchronous client applications automatically, it is hard to characterize the ratio of benign to attack messages, but in a typical 30 minute run, STEG handled thousands of legitimate messages and hundreds of attack-induced messages (i.e., attack messages were roughly 10% of the overall load in the system, which is fairly consistent with an adversary intending to cause damage without drawing undue attention). All attack requests were successfully mitigated by STEG. Fig. 8 shows a comparison of the latency of legitimate messages (IOs) when the system was facing no attacks (on the left) and under attack (on the right), aggregating the results from all attack runs. The figure shows that there was no significant impact on legitimate interactions during attack – the average latency of IO delivery is nearly identical. The attack runs showed a wider spread in latency values, but outliers as high as 1500 ms were observed in both runs.

Fig. 9 shows boxplots of the Mean Time to Mitigate (MTTM) for various STEG responses, measured as the latency between attack initiation and mitigation by STEG. Note that for attacks launched from the tactical side, MTTM includes the time it takes to transmit attack messages over the relatively disadvantaged 1 Mbps tactical network. The MTTM is low (~300 ms) for attacks involving incorrect TLS authentication and unauthorized HTTP, because they are dealt with early in STEG's execution pipeline. The largest amount of time (on the order of seconds) is spent in mitigating virus attacks (AV), followed by attacks involving identity mapping and unauthorized CoT messages, because their mitigation involves deeper inspection that happens later in the pipeline.

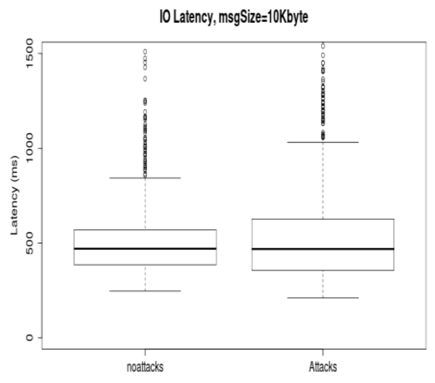


Fig. 8. Comparison of latency of legitimate IOs when under attack vs. no attacks

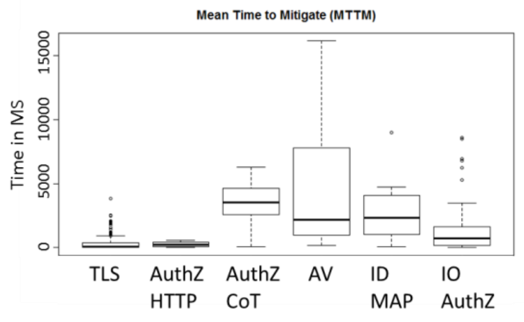


Fig. 9. MTTM for actions taken by STEG in response

C. *Assessing Performance Overhead*

To understand the performance overhead of STEG, we measured and compared the latency introduced by STEG in an unloaded situation. Since most of the interactions between environments are not possible without STEG, we constructed a representative baseline where the same amount of data is transported between two entities, one in the tactical network and the other in the enterprise network, through a pass-through forwarder using the *socat* utility [19]. We used a tactical producer publishing IOs to the enterprise IMS through STEG. To keep things comparable in the corresponding baseline, we used a program publishing IOs in SOAP format so that *socat* forwarding does not need to perform any protocol transformation in order for these messages to be accepted by the enterprise IMS.

The producer was configured to publish 10 KB IOs at the rate of one IO/second. To remove the effects of virtual networking which we observed to perturb the results, we ran these overhead experiments (both baseline and STEG) on physical machines. STEG, the enterprise IMS, and the enterprise producer ran on one physical node (representing the enterprise side). The tactical consumer ran on the other physical node (representing the tactical side). The physical node used to host STEG in these experiments was a 8 Core Intel(R) i7 CPU @ 2.70GHz with 32 GB memory running Fedora 18.

Table 1. IO latency STEG vs. baseline low load

	min (ms)	max (ms)	mean (ms)	std (ms)
Baseline	27	710	66	40
STEG	74	1737	113	100

Table 1 shows the results of this experiment. STEG introduces an average of 70% overhead (1.7×) in latency with higher variance (standard deviation). Considering that the baseline configuration that we used is an idealized one, i.e., there doesn’t exist any such connection from the enterprise to the tactical environment and we are evaluating an as-yet-unoptimized prototype, we believe that the overhead incurred is low for the benefit that STEG provides. Follow up experiments using different peeled-back configurations of STEG that are out of scope of this paper, showed that there are three major contributors to the overhead compared to the *socat* baseline: (a) plumbing, (b) security processing at the Crumple Zones, and (c) the dissemination processing, and they all behave differently as request load (i.e., the number of clients issuing requests at a constant rate) increases. The plumbing component (i.e., accepting connections and requests at the Façade) scales well and has a higher threshold for causing saturation. The security processing at the Crumple Zones is likely to saturate faster and cause incoming requests to back up, but can be mitigated by increasing the processing power of STEG. The dissemination processing is limited by the available tactical bandwidth, and if all messages have the same applicable QoS policy it starts to fill up its queues with messages it cannot discriminate, leading to cascading effects downstream.

V. Related Work

Gateways are often used as network access points (such as the default gateway of network connected computers) or as devices that connect two network segments with administrative or technical differences (e.g., gateways connecting provider networks in the Internet backbone). While network gateways are mostly about routing and protocol mapping, STEG focuses on security and QoS asymmetries, which includes protocol mapping but not routing.

Making Internet-resident web content available to smart phones and tablets is similar in spirit to the end-effect that STEG aims to achieve. Commercial vendors, e.g., *mobify* [16], adapt the content and presentation of existing websites for mobile devices and deliver them over a separate content delivery network. Software systems such as IBM WebSphere and Oracle/BEA WebLogic support “multi-channel content adaptation” [9][15]. In WebLogic, request attributes are used to determine the nature of the requesting device and the subsequent dispatch of the request to the appropriate handler. WebSphere requires that the content is created in a device-independent standard format that its multichannel engine can customize based on the originating device. These technologies focus primarily on content and presentation, which cover some QoS aspects but assume that mobile and non-mobile devices provide the same level of credentials expected by the portal, which may mean open access (e.g., cnn.com) or user name/password (e.g., Facebook.com). Another difference is that these technologies cover content delivery from enterprise to mobile only, often aided by special-purpose mobile apps.

Mission-critical operations often take place in resource restricted networks and in situations that lack or hinder connectivity. Research in *Marti* [12] shows how to enable beyond line-of-sight interoperation and manage the QoS between tactical users under these circumstances. STEG builds upon the QoS handling aspects of Marti to manage the QoS of information delivered to the tactical side. Mission operation also often entails information exchange between environments with different levels of security and control (e.g., classified vs. unclassified). Such interactions are facilitated by specialized Cross Domain Solutions similar to STEG but primarily concerned with enforcing the applicable security control. STEG does not handle multiple levels of security.

VI. Conclusion

STEG is intended for environments that include mobile devices serving mission-critical needs, perhaps using non-commercial radio networks, needing enterprise reachback for improved situation awareness, to offload processing, or to provide sensor inputs to enterprise services. Initial evaluation has shown the benefit of STEG to protect services and consumers in each environment against a wide variety of attacks originating from the other environment, and to manage the QoS provided to users, especially in the disadvantaged tactical environment. Ongoing work is enhancing the prototype in a variety of areas including adding support for multicast and additional interaction patterns. Larger scale experiments in realistic contexts are needed to establish operational feasibility and deployment-specific performance tuning.

REFERENCES

[1] Atighetchi, M., Pal, P., Adler, A., Gronosky, A., Yaman, F., Webb, J., Loyall, J., Sinclair, A., Payne, C., Crumple Zones: Absorbing Attack Effects Before They Become a Problem, CrossTalk – The Journal Of Defense Software Engineering, 24(2), pp. 4-11, March/April 2011.

[2] Atighetchi, M., Ishakian, V., Loyall, J., Pal, P., Sinclair, A., Grant, R., Metrinome – Continuous Monitoring and Security Validation of Distributed Systems, *Journal of Cyber Security & Information Systems,* to appear, February 2014.

[3] Atighetchi, M., Soule, N., Pal, P., Loyall, J., Sinclair, A., Grant, R., Safe Configuration of TLS Connections, Proceedings of the IEEE Conference on Communications and Network Security (CNS), 2013, pp. 415–422.

[4] Clam Antivirus, http://www.clamav.net/lang/en/, Accessed February 15, 2014.

[5] Dierks, T., Rescorla, E., The Transport Layer Security (TLS) Protocol Version 1.2, August 2008, RFC 5246, http://tools.ietf.org/html/rfc5246.

[6] Fossi, M., Egan, G., Haley, K., Johnson, E., Mack, T., Adams, T., Blackbird, J., Low, M.K., Mazurek, D., McKinney, D., et al, Symantec Internet Security Threat Report Trends for 2010, Volume 16, April 2011.

[7] Gillen, M., Loyall, J., Sterling, J., Dynamic Quality of Service Management for Multicast Tactical Communications, Proceedings of the 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2011, pp. 11–18.

[8] Gong, L., Mueller, M., Prafullchandra, H., Schemers, R., Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2., Proceedings of the USENIX Symposium on Internet Technologies and Systems, 1997, pp. 103–112.

[9] IBM Mobile Portal Accelerator, http://www-01.ibm.com/software/lotus/portal/value/mobileaccelerator/features.html. Accessed February 15, 2014.

[10] Kuhn, D., Coyne, E., Weil, T., Adding Attributes to Role-Based Access Control, *Computer.*, vol. 43, no. 6, pp. 79–81, 2010.

[11] Loyall, J., Carvalho, M., Martignoni III, A., Schmidt, D., Sinclair, A., Gillen, M., Edmondson, J., Bunch, L., Corman, D., QoS Enabled Dissemination of Managed Information Objects in a Publish-Subscribe-Query Information Broker, Proceedings of SPIE 7350, Defense Transformation and Net-Centric Systems, Orlando, FL, USA, 2009.

[12] Loyall, J., Gillen, M., Cleveland, J., Usbeck, K., Sterling, J., Newkirk, R., Kohler, R., Information Ubiquity in Austere Locations, *Procedia Computer Science*, Vol. 10, pp. 170–178, 2012.

[13] Loyall, J., Gillen, M., Paulos, A., Edmondson, J., Varshneya, P., Schmidt, D., Bunch, L., Carvalho, M., Martignoni III, A., Dynamic Policy-Driven Quality of Service in Service-Oriented Systems, Proceedings of the 13th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2010, pp. 1–9.

[14] Mobify homepage, http://www.mobify.com/solutions/developer/. Accessed February 15, 2014.

[15] Oracle, http://docs.oracle.com/cd/E13155_01/wlp/docs103/portals/multichannel.html. Accessed February 15, 2014.

[16] Rash, M., Single Packet Authorization, *Linux Journal*. April 01, 2007.

[17] Robbins, D., Unmanned Aircraft Operational Integration using MITRE's Cursor on Target, *The Edge*, Volume 10, Number 2, MITRE, 2007.

[18] Smalley, S., Vance, C., Salamon, W., Implementing SELinux as a Linux Security Module, NAI Labs Technical Report, February 2006.

[19] Socat Homepage, http://www.dest-unreach.org/socat/. Accessed February 15, 2014.